CSU11013 Programming Project 2024

1. Goal

The goal of the project is to construct an application to explore data relating to commercial flights. We will use a public dataset of one month of flights from the US Bureau of Transportation Statistics. The application will read in data from a file “flights.csv”, render it, and allow the user to interact with it. The data will take the format of *comma separated values*, which will contain a number of rows, each of which will describe one flight. Each flight will be described by data in a number of columns (separated by comma characters - ‘,’).

Each entry appears on a new line. Varying size datasets will be available from links on blackboard (537k, 100k, 10k, and 2k entries), and the ability to deal with these will depend on the efficiency of your program. You may make use of other datasets to add additional data such as country names. A full description of the fields, and the ability to download further months of data is available at
https://www.transtats.bts.gov/DL_SelectFields.aspx?gnoyr_VQ=FGK&QO_fu146_anzr=b0-gvzr

FlightDate – The date of the flight in yyyymmdd format – you may ignore the hours and minutes (which are zero).
IATA_Code_Marketing_Airline – Code to identify the carrier e.g. AA.
Flight_Number_Marketing_Airline – The flight number (combine with the carrier code to obtain the usual format).
Origin - Origin Airport (text string).
OriginCityName - Origin Airport, City Name (text string).
OriginState – Origin Airport, State Code (text string).
OriginWac – Origin Airport, World Area Code (integer, see separate lookup table to map numbers to countries).
Dest - Destination Airport (text string)
DestCityName - Destination Airport, City Name (text string).
DestState - Destination Airport, State Code (text string).
DestWac - Destination Airport, World Area Code (integer, see separate lookup table to map numbers to countries).
CRSDepTime – Scheduled departure time (hhmm)
DepTime – Actual departure time (hhmm)
CRSArrTime – Scheduled arrival time.
ArrTime – Actual arrival time.
Cancelled – Cancelled Flight indicator (1=yes). Diverted – Diverted Flight indicator (1=yes).
Distance – Distance between airports in miles.

Fields may be empty (e.g. cancelled flights), so it is best to program defensively.

2. Structure

The program will contain the following components:

- code to read in the data from a file and place it in classes.
    - Processing provides both `loadBytes` and `loadStrings` commands. The `split` method should also be helpful.
    - A simple (although not particularly efficient) solution would be to define a DataPoint class which represents a single flight. There would be one instance of the class for each entry in the input file.
    - You may wish to store the data in a format which is more efficient to access.
- code to select a subset of this data.
    - Not all the data will be shown on the screen at one time, and so a set of queries must be defined in your code. At a minimum, the following queries should be implemented:
        - Flights associated with a particular airport.
        - Flights within a certain date range
        - Flights sorted by lateness
- code to draw the data to the screen.
    - The results of each query will need to drawn on the screen.
    - You are encouraged to use graphical representations where appropriate (eg. some data could be on a barchart, a 2D or 3D representation of flights, or more complex visualisations such as heatmaps).
- code to handle user commands.
    - Selecting what data is to be displayed (the query), the airport, airline, date range, etc.

- code to put everything together
    - You are advised to have an outline of this as early as possible (first week of project).

3. Assessment
The project marks will be allocated according to both your individual effort and the group performance. This is a group project, and part of the project is to manage the group effort.
Each individual will receive a mark based on their Team's performance and their own contribution. **If you make no contribution to individual or group tasks, you will receive no marks**.

You will be required to submit and demonstrate the current status of your project **every week** until the end of the semester. **Half** of the project marks will be allocated for the **weekly progress**. DO NOT MISS THE LABS.

It is not possible to leave the project until the end of the semester.

Code must be maintained with the Git version control system with all team members making commits and sharing access with team members and demonstrators via GitHub. Every individual must submit the latest version of the code every week through Blackboard as well as a final submission before the deadline. All code must be commented, and the authors indicated. You **must** use comments to indicate revisions to the code (eg. " M. Jones, Added Graph class for displaying results, 8pm, 10/3/2024". "J. Smith, Updated to show the dates on the chart, 7pm 14/3/2024", "L. White, Fixed bug which stopped user from going back to homepage, 9pm, 15/3/2024". etc.). It is in your own best interest to get credit for the code you have written!

Attendance for the **full duration** of the lab session is **mandatory** and you will be expected to work with your team during the labs. Students who are significantly **late** or leave early will be given a **zero mark** for that session.
An allowance will be made for absence or lateness due to a documented illness or bereavement.

The project will take more time than is available at the labs. Remember that CSU11013 is marked **only on coursework**. You will need to schedule time in your groups outside of lab time to work on your assignment / divide up workload.

The assessment will focus on the demonstration of working code; a check on code authorship; a check on quality of code and documentation, progress towards the overall goal of the project, and the features implemented. Your code is to be accompanied by a report of **1000 words** (plus screenshots) outlining your design and any ways in which your solution goes beyond the original project brief. You must include a section detailing the contribution of each team member and include a full git history.

You will present your project with a 3 minute video and attend the lab in the last week of term for a Question and Answer session with a demonstrator.

4. Assessment Schedule
The following is the schedule for the project, including milestones to demonstrate:

Week 6	Project Handout	Plan project. Start thinking about it. Outline main program.
Week 8	File reading and code outline	Demonstrate reading of data from file and give outline of main program, including major classes, sketch of screens.
Week 9	Initial rendering of data	Graph and textual output of data.
Week 10	Data selection and rendering	Apply user queries to data set and present results on screen.
Week 11	User Interaction	Select different options and change screens
Week 12	(10th April) Deadline for Blackboard submission, code, report and video. (11th April) Final Lab Q and A	Full demo with 3 minute video and submission of report – **1000 words MAXIMUM**. Code submitted through Blackboard.

5. Assessment Weighting

The project is assessed throughout the second half of the semester and has a total weighting of 80% of the final mark.

Week	**Assessment Method**	**Weighting**
Week 8	Interview with a panel of demonstrators.	10%
Week 9	Interview with a panel of demonstrators.	10%
Week 10	Interview with a panel of demonstrators.	10%
Week 11	Interview with a panel of demonstrators.	10%
Week 12	Video Demonstration and Q and A, code and report.	20%
Week 12	Report and well commented code and version control.	20%

6. Frequently Asked Questions.

- You **may** use multiple tables. You may use additional datasets. You may pre-process your data into files.
- You may **not** use an alternative development environment (IDE) **without permission**. Such permission will **only** be granted if **all** team members indicate they are happy to do so, and I am convinced that all team members are fully engaged with the project.
- You **must** use the provided version control repository rather than any other revision control system. It is used to track individual contributions.
- If you are using SQL, you must implement a version which uses a local database (with a smaller dataset) so that your program can be checked independently. Your program **must** work when downloaded from the repository to a clean lab machine.